# Agents for enhanced end-host multicasting in distributed simulation systems

Prepared by:

Dr. Robert Simon

Dr. J. Mark Pullen

George Mason University

Fairfax, VA 22030

## SUMMARY

This report studies the use of distributed agents to improve the communication efficiency and effectiveness of XMSF-based simulations. We first present some basic architectural background on the use of agents with current work in Web Services and the Semantic Web. Using an existing multicast-enabled HLA simulation system as a starting point, we suggest an approach for "agentifying" this system using a minimal number of agents. We then describe how these agents initialize and configure themselves into an interconnection pattern and introduce several novel algorithms for agent interconnection.

We experimentally evaluated our approach, using two different heuristics, against IP Multicast and pure unicast. Except in the circumstances of extremely dense group membership, our approach performs in a competitive fashion to IP multicast. By placing intelligent agents for packet forwarding duties on end-hosts it is possible to actually *reduce* the computational burden within the system. Our experiments show that with the proper choice of agent placement algorithms a large-scale agent-based simulation system can be effectively supported by end-host multicast.

## Introduction

One of the problems currently facing distributed simulation system designers is how to best take advantage of web-based technologies within the evolving XMSF framework to provide for simulation service interoperability, composability, extensibility and discovery. Since many distributed simulation systems targeted under the XMSF initiative require real-time performance, a closely related problem is how web-enhanced simulation environments can effectively use network level Quality-of-Service (QoS) options for reliability, bounded latency and group-based multicast. Further, we expect that software agents will play a central role in XMSF simulation deployment strategies. A challenge in addressing agent-oriented XMSF-based simulation technologies and network-level control policies is that they operate at different system levels, and the policies and semantics at each level do not readily translate into efficient support at another level. One specific issue critical to the successful deployment of large-scale distributed virtual simulation environments is that simulation agents may not make efficient use of available network multicasting technologies such as end-host multicast, whereby the multicast function is shifted from the network into the application [3]. End-host multicast is required when IP multicast is not available. The focus of this report is the design and evaluation of an agent-based architecture and set of interconnection algorithms to efficiently use end-host multicast in the Next Generation Internet (NGI).

The starting point for this study is our previous work in web-based distributed virtual simulation (DVS) system over the NGI [10]. Portions of this earlier work described our NGI Internet Federation Object Model (NGI FOM) and the Selectively Reliable Multicast transfer Protocol (SRMP). The NGI FOM provided a structured FOM to support a variety of multicasting styles within a distributed simulation system such as the HLA [15]. SRMP is a transport level protocol that supports a mix of reliable and best-effort multicast by taking advantage of the specific requirements of the DVS. For instance, many data streams are refreshed frequently and thus may be distributed by best-effort transport, while a small fraction of traffic consists of data that changes rarely and thus requires reliable transport.

Our original work was implemented by modifying an RTI supported by TAO. The goals of this study are to describe a set of web-based agents capable of supporting large-scale distributed simulation systems and to evaluate their use within an overlay multicast environment. Specifically, this report studies the possibility of extending our previous approach to take advantage of several proposed web-based technologies that fall within the broad scope of the XMSF initiative, especially distributed agents using Web Services [5] and the Semantic Web [9]. We first describe an "agentified" version of our work in the NGI FOM and SRMP, and suggest how to use them to effectively support an end-host DVS application. We next design and evaluated interconnection algorithms between agents that make efficient use of the network. Our evaluation approach is through discrete event simulation to determine the performance, in terms of delay, throughput and resource utilization, of the interconnection algorithms within a Wide Area Network for different numbers of agents.

## Background

The use of agent-based computing in designing flexible and extensible simulation systems has long been recognized [1], and is one of the motivating factors for XMSF. Agents are a powerful software engineering unit that can encapsulate complex rule sets, can be composed from other agents and can communicate semantically rich messages via agent communication languages. Messages passed between agents are described by knowledge representation schemas called ontologies [5].

Agents in the support of simulation systems are particularly well suited for deployment in the context of recent work in Web services and the Semantic Web. Web services are applications or software components that are deployed on the World Wide Web. Web services have an associated suite of enabling protocols, such as the WSDL [8] (Web Services Definition Language). Web simulation services must be described along multiple dimensions, including both syntactic (how to use it) and semantic (what does the service mean and what does it offer). While methods for syntactic descriptions are

generally well understood for distributed simulation, the issue of semantic description is still considered an open question. One promising approach to this problem is through ongoing efforts in Semantic Web service description, such as DAML-S [4].

A potential drawback in using web-enabled agent-oriented simulation is that system performance will suffer. This is particularly true in the case of group-based communication systems that use end-host multicast, where group management and packet replication is performed on end host systems, rather then within the network, as in the original IP multicast model. Multicasting has long been recognized as an essential element for the success of large-scale distributed virtual simulation systems. As the end-host multicast model increases in importance, there is growing concern about its impact on system performance, especially for systems such as time sensitive DVS. Since agent composition and location can cause communication bottlenecks, it is important to design effective agent placement algorithms for end-host multicast. Our approach for addressing this problem is to first define a generic, general purpose agent architecture and then to propose and evaluate a set of agent communication interconnection algorithms.

## Proposed Agent Architecture

As explained in the introduction, our starting point is our previous work in supporting HLA and DIS over the NGI Internet Federation Object Model (NGI FOM), via the Selectively Reliable Multicast transfer Protocol (SRMP). The NGI FOM provided a structured FOM to support a variety of multicasting styles, including end-host multicast. In order to agentify this distributed simulation environment we must strike a balance between functionality and simplicity. The tradeoff is between the number of agent types deployed within the system versus the complexity of managing and controlling agent interaction. As a first step, we propose three basic agent classes. The first class is *federate* agents, and is used to represent and control individual simulation programs associated with a particular NGI FOM. The function of these agents is to allow for automatic service discovery, service invocation, service composition and service execution monitoring for simulation federations. The second agent class is *fed_com* agents, short for federation communication. These agents are used to specify and control

communication level reliability and group communication issues, such as multiple types of reliability for different data objects in the same federation by providing agent control over SRMP functionality. The third agent class is *serv_comp* agents, short for service composition agent. These agents interact with the multicast service to coordinate the construction of multicast overlay trees. Serv_comp agents determine the interconnection pattern between fed_com agents. They also possess the necessary functionality to perform data conversions, such as lowering the data rate to certain members of a group, by means of compression techniques, data distribution management, etc.

These three classes of agents must be able to communicate using a common and extensible ontology markup language. One powerful approach is to have agents communicate by exchanging web pages, with each page containing ontological information. This information can be created using DAML-S. In the DAML-S model, each ontological framework consisting of three subonotologies: a service profile, a process model and a grounding. The agent service profile describes what the service can do, for purposes of service advertisement, service discovery and service matchmaking. Matchmaking is the act of matching services with other services. The process model describes how a simulation service allows for control and data flow, in terms of inputs and outputs. The process model also specifies the service's necessary pre-conditions and side effects. The grounding specifies the details of how an agent can access a service. For instance, we can use WSDL as the format to specify fed_com communication modes and associated transport-level port numbers.

**Simulation agent ontologies.**

Here is a representation of what a small part of the profile portion of the serv_comp ontology description would look like. It offers two types of services, shown in bold face; one for native IP multicast and one for overlay multicast service. The text file storing this description is treated within in the system as any other web page. It can be stored, retrieved, searched for, advertised, etc. An agent would retrieve this page during the construction of a multicast group. It could then search for the type of multicast service, either overlay or native. The agent selects the service type and obtains

more detailed ontological descriptions of what is required to establish the service from information contained within the page.

<daml:Ontology rdf:about="">

<daml:imports rdf:resource="http://example.com/Service.daml" />

<daml:imports rdf:resource="http://example.com/MCServiceProfile.daml" />

<daml:imports rdf:resource="http://example.com/MCServiceProcess.daml" />

<daml:imports rdf:resource="http://example.com/MCServiceGrounding.daml"
</daml:Ontology>

- <**service:Service rdf:ID="MCOverlayService**">

<service:presents
rdf:resource="http://example.com/MCProfile.daml#Profile_Multicasting_Service" />

<service:describedBy
rdf:resource="http://example.com/MCProcess.daml#MCProcessModel" />

<service:supports
rdf:resource="http://example.com/MCGrounding.daml#MCGrounding" />

</service:Service>

- <**service:Service rdf:ID="MCNativeService**">

<service:presents
rdf:resource="http://example.com/MCProfile.daml#Profile_MC_Service" />

<service:describedBy
rdf:resource="http://example.com/MCProcess.daml#Native_MC_ProcessModel" />

<service:supports
rdf:resource="http://example.com/MCGrounding.daml#Native_MC_Grounding" />

</service:Service>

**Agent interconnection pattern**

Our approach defines two communication paths through the system. The first is direct agent-to-agent communication, and is used for setup and control purposes. This occurs as agents exchange web pages via an agent communication language (ACL). The second type is used directly for data transfer, bypassing the ACL. The reason for two paths is that agent communication methods and ACLs typically operate by exchanging XML messages. This technique is too slow for time sensitive data transport. The serv_comp agents also participate in data transfer, using direct network connections, without the use of an ACL.

A simulation is initiated when a federate agent finds other federate agents via the use of a system matchmaker. Under the direction of the initiating agent, the agents perform an all-to-all exchange of web page descriptions, which include their FOMs and requirements for runtime simulation parameters, based on the RTI model for simulation specification. Using this shared information, the runtime simulation requirements are encoded in the form of a XML description. Each federate agent then discovers a serv_comp agent to determine network connectivity and data management requirements. This includes a specification of the federate's required data rate and the selection of the desired multicast service, either IP Multicast or end-host.

Serv_comp agents manage multicasting and serve to support both heterogeneous data rates and data models within the system. Data rates are determined by network connectivity and application needs. For instance, in the case of DVS systems requiring streaming data or media streams different data rates can be supported. This need may arise from the use of different compression technologies or different levels of network connectivity. Interest Management techniques in the HLA also can result in different required data rates. The data model is a function of the FOM.

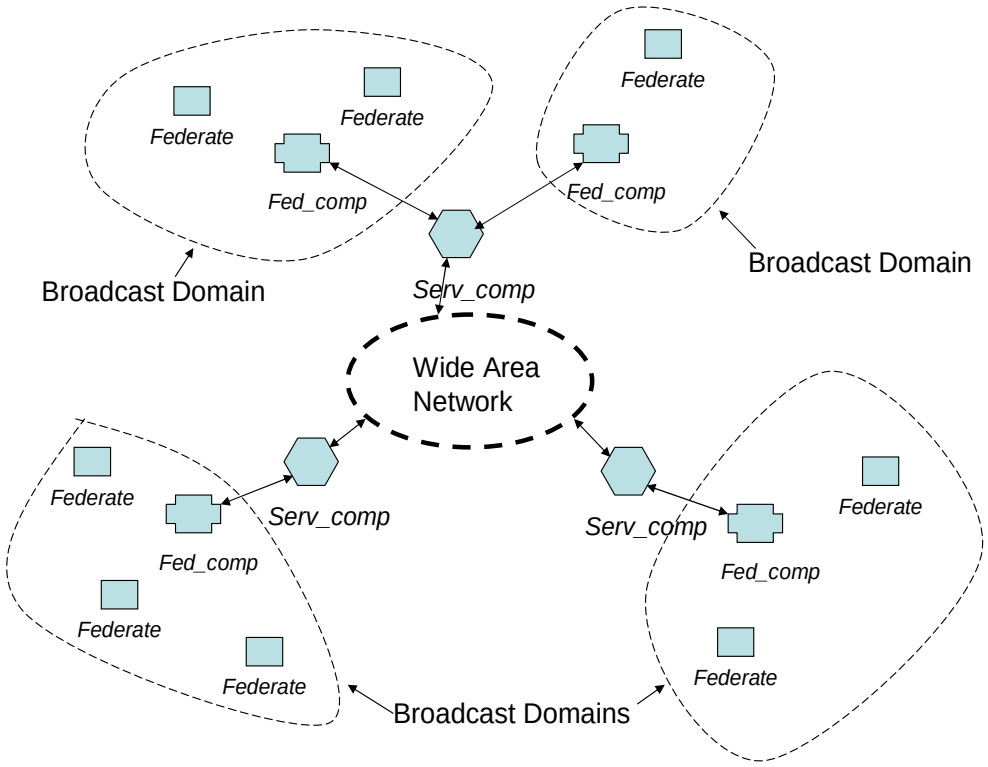


**Figure 1: Logical simulation agent architecture**

At the level of system architecture, the three agent types coordinate their actions to manage and maintain overlay multicast communication. Fed_comp agents manage local domains that represent connection points between federate agents running on end-hosts and the Wide Area Network (WAN). A local domain may be one or more Local Area Networks (LANs), a single high-end computer connected directly to a router, etc. We characterize the local domain as a broadcast domain. In effect, each broadcast domain is the end-point of a logical multicast distribution tree.

Based on the simulation specification and the location of federates inside each broadcast domain, each serv_comp discovers and assigns fed_comp agents to provide transport level mixed data delivery mode support. Thus, the number and location of each fed_comp agent represents the end-points in the logical data distribution tree. A logical overview of this architecture is shown in Figure 1. If native IP Multicast were possible, fed_comp agents could be placed directly in a spanning tree. However, under the end-host scenario each fed_comp agent needs to perform data forwarding and packet

replication. Each fed_comp also may need to perform data management actions, for reasons such as data distribution management. Therefore, the interconnection pattern between fed_comp agents is of central importance to system performance.

The purpose of defining and analyzing fed_comp interconnection algorithms is to reduce communication latency and system load, thus improving end-to-end performance. In order to manage the overlay, all end-host overlay techniques cause extra packet transmissions over the network. Our goal is to minimize this effect and reduce actual network utilization. Our approach to multicasting is single-source tree construction. That is, each sender is the root of a tree that consists of all the receivers. Further, we dynamically place serv_comp agents within the tree at specific places in order to reduce data traffic transmission, as explained below.

The algorithm has three phases. In phase one, we create a mesh of fed_comp agents ordered by data rate. In phase two, we create a series of single source overlay multicast trees. For phase two, we add application-level admission control, in an attempt to limit congestion over any particular portion of the WAN. We also use a load-balancing path selection heuristic. Our algorithm is flexible, and uses several different approaches to tree creation. In phase three, we place extra serv_comp agents in the system to support the data rate of receiving fed_comp agents, as required. This allows us to further reduce bandwidth utilization. The details follow.

**Phase 1:** Create a mesh of *fed_comp*s for each different source data rate. The mesh is a complete virtual graph whose vertices are the *fed_comps*. First, find the shortest path between each *fed_comp* using link delay as the link cost. Then add a uni-directional virtual link between the two group members, (i.e. *fed_comp*) to the mesh, *only* when 1) there is a path that starts from a sender to a receiver *fed_comp*, 2) the data rate requested by the sender is greater than or equal to the data rate requested by the receiver, and 3) the physical links of the shortest path have the available bandwidth greater than or equal to the data rate requested by the receiver.

We repeat this procedure to create a mesh per each source's different data rate. If the sources' data rates are the same, a mesh can be used to create overlay multicast trees rooted on the sources with the same data rate.

**Phase 2:** Create a source-based overlay multicast tree over the mesh that the source's data rate matches. There are several novel approaches during this phase. We do application-level admission control, insuring that no particular link or path is overloaded. Our approach accommodates either a shortest path or a spanning tree routing construction. Further, we use a load balancing heuristic based upon avoiding paths that are unevenly utilized.

During construction of the source-based multicast tree, at each step, a *largest data rate edge* connecting an off-tree node to the partial tree is added to the tree. Let $P_{\hat{m}_i,\hat{m}_j}$ be the path between group member $\hat{m}_i$ and group member $\hat{m}_j$. When a node decides its parent, it chooses the node connected with the mesh link that has the least value of $(1 - ILBR(P_{\hat{m}_i,\hat{m}_j}))$, where $ILBR(P_{\hat{m}_i,\hat{m}_j})$ (*Independent Least-balanced Routing* [12]) is defined as

$$ILBR(P_{\hat{m}_i,\hat{m}_j}) = \frac{\prod_{\forall z \in P_{\hat{m}_i,\hat{m}_j}} ec_z}{\sum_{\forall z \in P_{\hat{m}_i,\hat{m}_j}} ec_z}$$

The excess capacity $ec_z$ at each link z is defined as the available bandwidth as a percentage of total link capacity of the underlying network layer links. This information is obtainable via standard network monitoring and measurement tools. The purpose of using the ILBR technique is to favor shorter paths over longer paths, provided that the shorter paths are not excessively congested.

Only when the available bandwidth of the bottleneck link of the path between two group members supports the receiver's data rate is the path added to the overlay multicast tree. If a path is added to the overlay multicast tree, the available bandwidth of each link on the path is reduced by the receiver's data rate. Each parent *fed_comp* keeps only the *fed_comps* with the highest data rate request among the children candidates as its

children. The reason for this is to try to make *fed_comps* with similar data rates as children of the same parent, leading to a reduction of the number of *serv_comp* agents in the overlay multicast tree. The algorithm is given is full detail in Appendix A.

**Phase 3**: Place *serv_comps* to adapt the data rate to users' requests at the nodes on the overlay multicast tree that have the children whose requested data rates are smaller than the node's. Finally, at the end of Phase 3, direct network connections are instantiated between the *fed_coms* in the system, along with connections between any added *serv_comps* and their associated *fed_coms.*

The result of the three phases of this algorithm is direct transport level network connections between *fed_comps,* the agents that control SRMP functionality. Further, if as a result of Phase 3 additional *serv_comp*s were added for data reduction, there are transport level connections setup between these *serv_comps* and other *fed_comps* in the system. Recall that for data transfer all connections use regular transport level protocols, not agent communication protocols.

## EVALUATION

We have conducted an experimental evaluation of our approach for constructing fed_comp interconnection patterns. We compare the performance of the Shortest Path-based heuristic and the Minimum Spanning Tree-based heuristic with the performance of IP Multicast, using DVMRP, along with pure unicast. We determine the IP Multicast tree based on the unicast paths from the source to each receiver. The evaluation metrics for the performance of each algorithm includes the number of serv_comp agents needed to be placed for a simulation, the average link stress, the worst link stress, the resource usage, the total bandwidth usage, and the relative delay penalty (RDP) of each path.

In reference [6], the stress of a physical link is defined as the number of identical copies of a packet carried by a physical link. For our evaluation, the physical link stress

is the ratio of the number of packets that a link processed to the number of packets sent by all members. Resource usage is the sum of the product of the link stress and the link delay over all links used. We assume that links with higher delay values tend to be associated with a higher cost metric. The total bandwidth usage is the sum of all bandwidth reserved along the paths from all sources to all receivers. The resource usage and total bandwidth are metrics to measure the network resources consumed in the process of data delivery to all receivers. RDP is the ratio of the latency experienced when sending data using the overlay to the latency experienced when sending data directly using the underlying network.

In our simulation, all fed_comp agents send and receive packets. We created a separate source-based multicast tree for each member to send out packets to all other multicast group members in IP Multicast, the Shortest Path-based overlay heuristic and the Minimum Spanning Tree-based heuristic. The admission control is performed when a path is added to the multicast trees in IP Multicast, the shortest-tree-based heuristic and the minimum-tree-based heuristic, and also for a path between the source and receiver for uni-cast. A source-based tree is accepted only when there is capacity enough to meet the data rate requested by each member for all the paths from a source to all other multicast group members. This capacity knowledge is derived from standard network measurement techniques. Otherwise, the tree is rejected.

We used the CSIM package to write the simulation programs, and Georgia Tech Internetwork Topology Models (GT-ITM) [14] to generate transit-stub graphs with 100 nodes. The multicast group size varies from 5, 10, 15, 20, 25, and 30. In this model 30 group members implies that 30% of the routers are part of the group, which is quite a high percentage compared to a typical WAN scenario. We included this worst-case scenario as a stress test for our approach. The members are uniformly assigned to each node on the graph. We ran the simulation program for each heuristic 30 times per group size. Each time, we used a different graph with 100 nodes. We used the number for each link in the graph generated by GT-ITM as the link delay. The delay of the local link is randomly distributed between 1 and 15. We used 1 Gbps for the bandwidth of the backbone links and 20 Mbps for the local link between the router and the end host. The

data rates requested by the fed_comp members are randomly distributed among 150 Kbps, 60 Kbps, and 20 Kbps.

The first set of results in Figure 2 shows the effect of data rate reduction, by comparing the number of serv_comp agents placed in the two overlay techniques to what would happen in IP Multicast. In the latter case, data reducing gateways are required on the routers of the IP multicast tree. This comparison is not required in unicast since in the unicast case the all streams are sent separately.

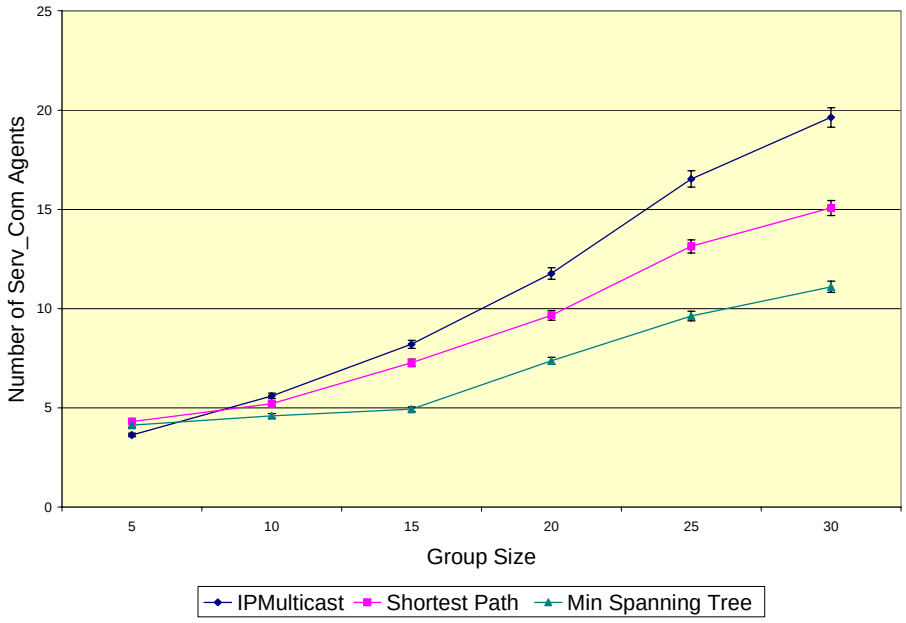


Figure 2 The number of serv_comp Agents due to data reduction

For the runs with the maximum number of group members (30), the range of the averages is between 9.98 and 10.42 for the Minimum Spanning Tree-based heuristic, between 15.86 and 16.04 for the Shortest Path-based heuristic, and between 19.23 and 20.04 for IP Multicast at 95% confidence level. We found that the overlay approach actually reduces the number of data-reducing agents required over IP multicast, which must be modified to make this type of intelligent data reduction in the data path. This is a strong indication that agent mobility is important.

Figure 3 shows average link stress. As above, for runs with the maximum number of group members (30), the range of the averages is between 0.397 and 0.401 for IP Multicast, between 0.80 and 0.81 for the Minimum Spanning Tree-based heuristic, between 0.99 and 1.0 for the Shortest Path-based heuristic, and between 1.05 and 1.07 for unicast at 95% confidence level.

Figure 4 shows the maximum or worst case link stress for the scenarios. This gives us an indication as to the "imbalance" of the various approaches. The minimum-spanning-based heuristic has the larger worst link stress than IP Multicast, but has smaller worst link stress than unicast. However, the Shortest Path-based heuristic has the larger worst link stress then IP Multicast and unicast.

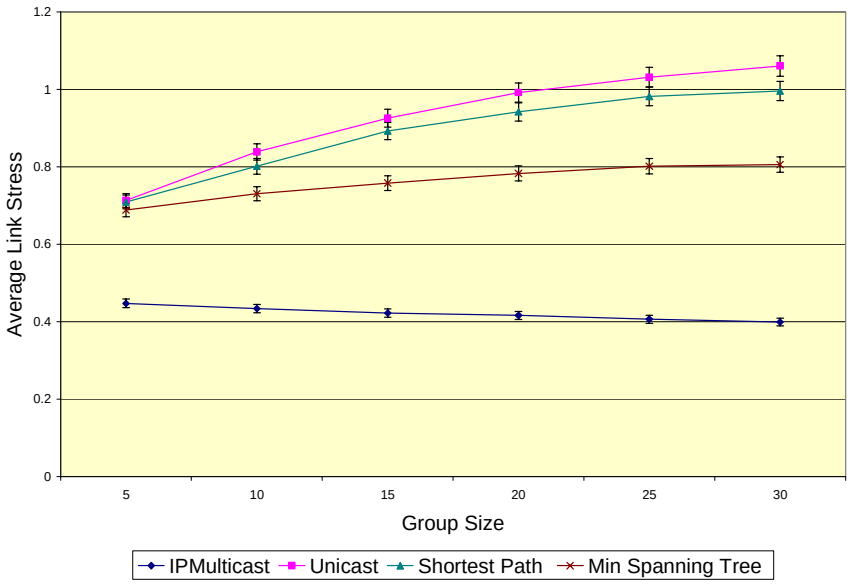


Figure 3 Average Link Stress

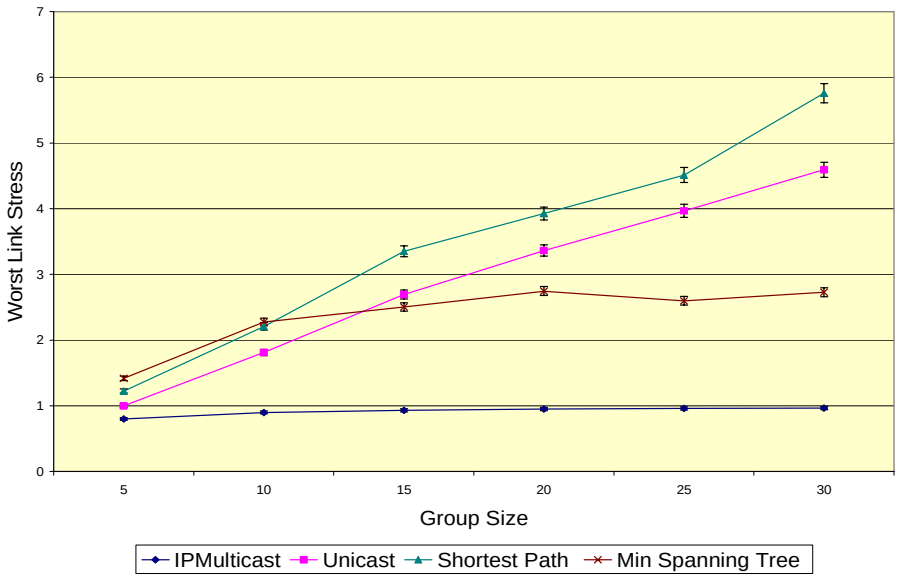


Figure 4 The Worst Link Stress

Figure 5 shows the total network capacity usage. The Shortest Path-based heuristic and the Minimum Spanning Tree-based heuristic both have larger total bandwidth usage than IP Multicast, but smaller then unicast. The Minimum Spanning Tree-based heuristic has the smaller total usage than the Shortest Path-based heuristic. The results show that our approach scales in a roughly linear fashion to the Multicast approach

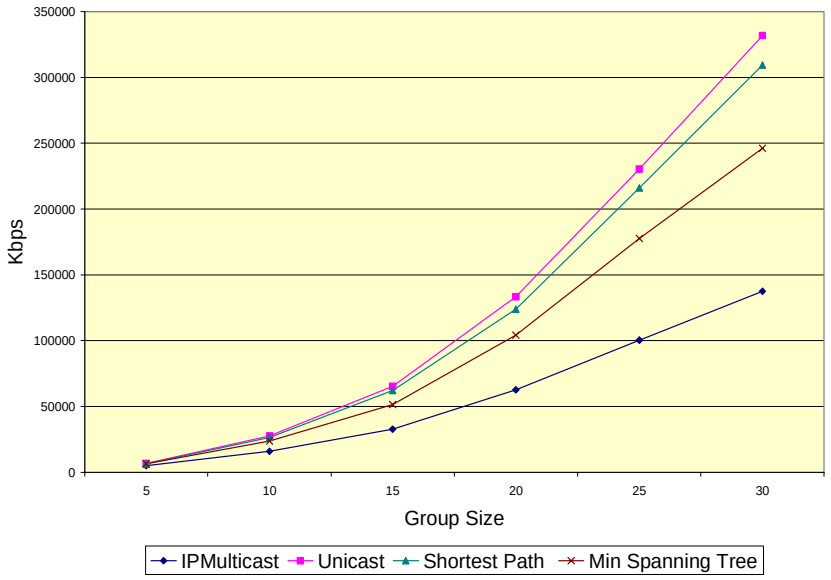


Figure 5 Total Capacity Usage

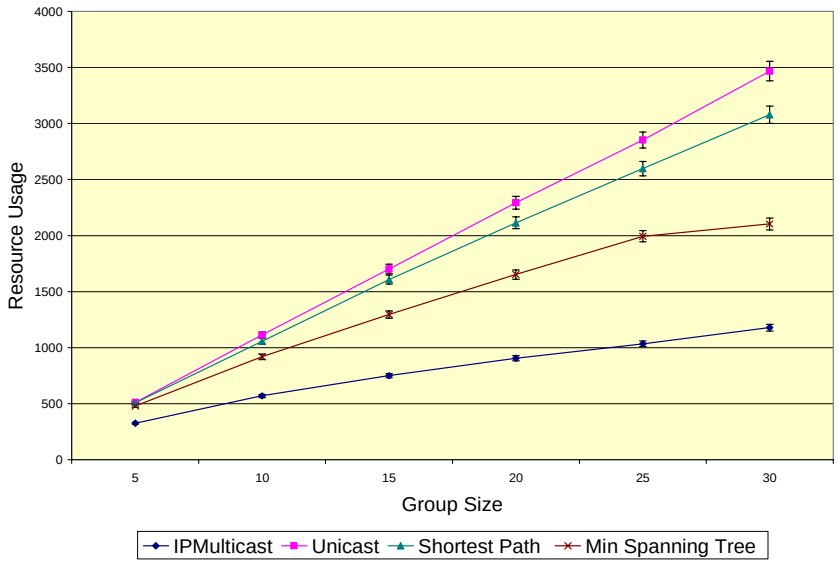


Figure 6 Resource Usage

Figure 6 shows Resource Usage. The Minimum Spanning Tree-based heuristic and the Shortest Path-based heuristic both have the larger resource usage than IP Multicast, but smaller than unicast. Once again, we see that our approach scales in linear fashion.

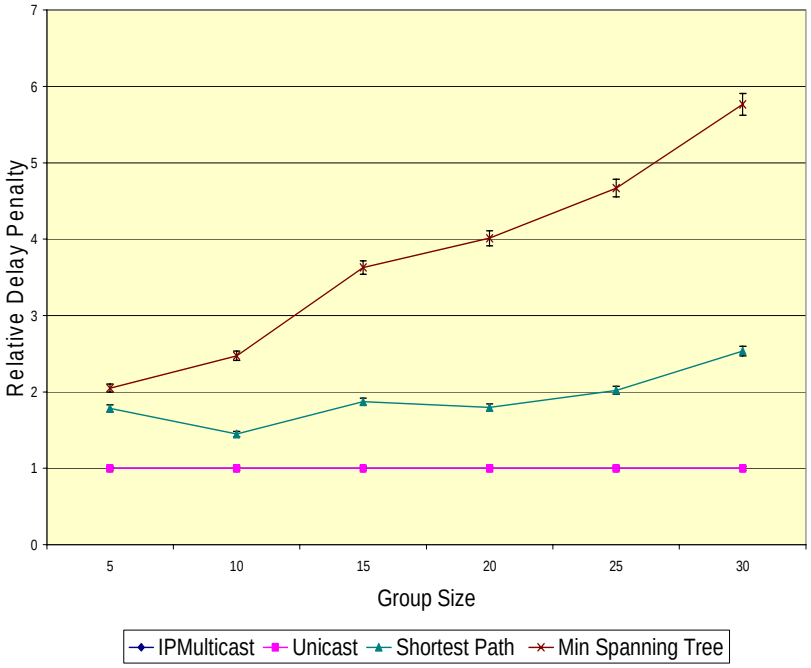


Figure 7 Relative Delay Penalty

Finally, Figure 7 shows the RDP. The Shortest Path-based heuristic has the smaller RDP than the Minimum Spanning Tree-based heuristic. We used 1 for the RDP of unicast and IP Multicast. The RDP penalty for the Shortest Path heuristic is quite competitive as compared to IP multi-cast and unicast, indicating the viability of our approach for time sensitive communication.

In summary, we see that for moderate group sizes, the overlay techniques are within a factor of 2 as compared to IP multicast for expected delay, stress, resource usage and bandwidth usage. When group members approach 30% of the available network

router size, overlay techniques are less efficient. However, in virtually all cases the overlay techniques outperform IP multicast for number of required data reduction agents. We observe that for networks where IP multicast is not available it is possible to still obtain good network performance, except in highly saturated networks. Further, since the end-host techniques can make intelligent use of packet rebroadcast, the number of data reducing agents required is reduced in the overlay methods, as compared to IP multicast.

## Conclusions and future work

This report described an agent-based architecture and interconnection strategy for a distributed virtual simulation environment. Using the broad framework of the XMSF initiative and our previous work in HLA and SRMP, we considered a web-based ontology for three classes of agents required to make a previously deployed system compatible with current work in Web Services and the Semantic Web. We then described how these agents initialize and configure themselves into an interconnection pattern. We introduced a new algorithm for agent interconnection, involving three separate phases. The outcome of the third phase involved setting up direct network connections, and may also result in additional agents being placed for purposes of data reduction.

We experimentally evaluated our approach, using two different heuristics, against IP Multicast and pure unicast. Except in the circumstances of extremely dense group membership, our approach performs in a competitive fashion to IP multicast. Further, the ability to place intelligent agents for packet forwarding on end-hosts actually reduces the computational burden within the system.

There are several natural extensions to this work. First, one intriguing result is that under some circumstances the overlay approach actually *reduces* the number of data-reducing agents required over IP multicast. The reason for this is that IP multicast must first be modified to make intelligent data reduction in the data path. This is a strong indication that agent mobility is important. For instance, an intelligent mobile agent could

dynamically discover data distribution patterns and use this information to prune and consolidate the multicast tree. However, the tasks of mobile agents within an XMSF framework are not well understood, and the issue should be investigated further.

Second, it is possible, if not likely, that future distributed virtual simulation systems will be deployed over networks that have a mixture of native multicast and end-host multicast. The interaction between these two multicast forms is not well understood, and therefore it would be worthwhile to extend our proposed algorithms to this environment.

Finally, it is also likely that some distributed virtual simulation environments will operate over a mixture of wireless and wired networks. In a wireless environment real-time applications face a number of challenges, including excessive packet loss due to transmission impairments and route breakages due to node mobility. XMSF-based agents should be able to be deployed in such environments, but before this happens the effect of a wireless network must be better understood. One possibility is to explore how agents could proactively mitigate negative wireless transmission effects via a combination of pre-caching, redundant information encoding, and mobility.

## REFERENCES


1. Agent Based Simulation 2002: 3rd Agent Based Simulation Workshop, Edited by Christopher Urban, Society for Computer Simulation, April 2002.

2. S. Chandrasekaran, et. al., G, "Web Service Technologies and their Synergy with Simulation," Proceedings of the 2002 Winter Simulation Conference (WSC'02), December 2002, pp. 606-615.

3. Y. Chu, S. Rao, and H. Zhang. "A Case for End System Multicast," Proceedings of ACM Sigmetrics, June 2000.

4. DAML: The DARPA Agent markup Language Homepage, available at http://www.daml.org.

5. J. Hendler, "Agents and the Semantic Web" IEEE Intelligent Systems, March 2001, pp. 30-37.

6. S. Jain, et. al., "A Comparison of Large-Scale Overlay Management Techniques," Technical Report UW-CSE 02-02-02, Computer Science and Engineering, University of Washington, 2002.

7. A. Kumar, et. al., "The Abels Brokering Sys-tem," 35th Annual Simulation Symposium, April 2002, pp. 54-62.

8. U. Ogbuji, "Using WSDL in SOAP applications," available at http://www-106.ibm.com/developerworks/webservices/library/ws-soap/index.html?dwzone=ws

9. S. A. McIlraith, D. L. Martin, "Bringing Semantics to Web Services," IEEE Intelligent Systems, January/February 2003 (Vol. 18, No. 1) pp. 90-93.

10. Pullen, J.M, Simon, R., Khunboa, C., Parupalli, M. and Brutzman, D., " Next-Generation Internet Federation Object Model for the

HLA," Sixth IEEE International Work-shop on Distributed Simulation and Real Time Applications (DSRT), October 2002, pp. 43-39.

11. A.P. Sheth and John A. Miller, "Web Services: Technical Evolution yet Practical Revolution?" IEEE Intelligent Systems (IEEEIS), Vol. 18, No. 1 (January-February 2003) pp. 78-80. IEEE Computer Society Press.

12. R. Simon and A. Sood, "Load-balanced Routing for Collaborative Multimedia Communication", 6th International Symposium for High Performance Distributed Computing (HPDC '97), August 1997, pp. 81 - 90.

13. S.J. Taylor and R. Sudra, "Modular HLA RTI services: the GRIDS approach," 35th Annual Simulation Symposium, April, 2002, pp. 15-22.

14. M. Thomas and E. W. Zegura. "Generation and Analysis of Random Graphs to Model Internetworks," Technical Report GIT-CC-94-46, College of Computing, Georgia Tech, 1994.

15. K., Weathery, R., and Dahmann, J., Creating computer simulation systems, Prentice Hall, New Jersey, 2000.

## APPENDIX A: Algorithm for overlay multicast tree creation

Let $\hat{M}$ be the group of *fed_comps* and $\hat{m}_i \in \hat{M}$ be a multicast group member.

Let L be a list of fringe *fed_comps.* The status of $\hat{m}_i \in \hat{M}$ can be UNSEEN, FRINGE, and IN-TREE. UNSEEN means the node is never visited, FRINGE means the node was visited, and IN-TREE means the node is added to the multicast tree.

Set the status of all $\hat{m}_i \in \hat{M}$ to UNSEEN;

Set source_node's status to IN_TREE;

current_node = source_node;

For all $\hat{m}_i \in \hat{M}$

{

For all nodes $w_i$ adjacent to the current_node

{

next_node = $w_i$ ;

If (next_node.status is UNSEEN)

{

/* admission_control reserves the bandwidth for
the path between current_node and next_node.
It returns true if the available bandwidth of the
bottleneck link of the path from current_node
to next_node is greater than or equal to the
next_node's data rate, otherwise returns false.
*/

If (admission_control(current_node, next_node)

== true)

{

next_node.status = FRINGE;

next_node.cost=

$1 - ILBR(P_{current\_node, next\_node})$;

next_node.parent=current_node;

Add next_node into the list of fringe nodes, L;

}

}//end of If

If (next_node.status is FRINGE)

{

new_cost = $1 - ILBR(P_{current\_node, next\_node})$ ;

If (next_node.cost > new_cost)

{

release_path_bandwith(next_node.parent,

next_node);

If(admission_control(current_node,

next_node) == true)

{

next_node.cost = new_cost;

next_node.parent=current_node;

}

```
                else
                {
                    //reserve the bandwidth again for the path
                    //between next_node and its parent.

                admission_control(next_node.parent,
                                        next_node);
                    }
            }          else if (the parent of next_node already has a
                        child whose data_rate is greater then that
                        of next_node)
                {
                    release_path_bandwith(next_node.parent,
                                            next_node);
                If(admission_control(current_node,
                   next_node) == true)
                {
                    next_node.cost = new_cost;
                    next_node.parent=current_node;
                     }
                     else     admission_control(next_node.parent,
next_node);
                 }//end of Else If
              }//end of If
           }//end of For
```

Add the off-tree node with *the largest data rate* among the fringe nodes to the multicast tree (IN-TREE);

Remove the selected node from the fringe list, L;

current_node = the selected node;

}//end of For